Association for Information Systems
AIS Electronic Library (AISeL)

MCIS 2010 Proceedings

Mediterranean Conference on Information Systems (MCIS)

9-2010

SPARSITY HANDLING AND DATA EXPLOSION IN OLAP SYSTEMS

Ina Naydenova
University of Sofia, Bulgaria, naydenova@gmail.com

Kalinka Kaloyanova
University of Sofia, Bulgaria, kkaloyanova@fmi.uni-sofia.bg

SPARSITY HANDLING AND DATA EXPLOSION IN OLAP SYSTEMS

Ina Naydenova, naydenova@gmail.com

Kalinka Kaloyanova, kkaloyanova@fmi.uni-sofia.bg

Faculty of Mathematics and Informatics, University of Sofia, Bulgaria

Abstract

A common problem with OnLine Analytical Processing (OLAP) databases is data explosion - data size multiplies, when it is loaded from the source data into multidimensional cubes. Data explosion is not an issue for small databases, but can be serious problems with large databases. In this paper we discuss the sparsity and data explosion phenomenon in multidimensional data model, which lie at the core of OLAP systems. Our researches over five companies with different branch of business confirm the observations that in reality most of the cubes are extremely sparse. We also consider a different method that relational and multidimensional severs applies to reduce the data explosion and sparsity problems as compression and indexes techniques, partitioning, preliminary aggregations.

Keywords: olap, sparsity, explosion, aggregates.

1 INTRODUCTION

OnLine Analytical Processing (OLAP) database systems have been studied for more than 15 years. They provide an easy-to-use interface for decision makers to navigate through their data. The modeling of such databases and the operations that can be applied are now well defined, and implemented by the main editors of database management systems as Oracle, Microsoft, IBM and others (Plantevit & Laurent & Laurent & Teisseire & Choong 2008). However the issues related with the optimization of physical storage and query performance continue to engage the interests of the scientific community (Wu & Shoshani & Stockinger 2010, Wu & Jiang & Ooi & Tan 2009, Bellatreche & Woameno 2009, and others). This is because the most OLAP applications are intended for interactive use, so people expect to get a fast response to queries (Pendse 2005). An independent Netherlands research has shown that end-users assume that a process has failed if results are not received with 30 seconds, and they are tend to reboot the computer, unless the system warns them that the report will take longer (Pendse 2008). Fast response is simple enough if queries merely have to retrieve information from a database, reformat it and present it to a user, but gets slower and more complicated if a significant amount of calculations have to be done to service the query (Pendse 2005). Important technique used to speed up the query response are preliminary aggregations, but the full pre-calculation approach fails with very large, sparse applications as the databases get too large (the database exponentially explode) whereas doing everything on-the-fly is too slow with large databases.

Data explosion is a key issue in most of the OLAP server environments. It is closely related with the sparsity phenomenon in multidimensional model - a relatively high percentage of the data points of the multidimensional space contain no value. In this paper we consider different methods that OLAP sever applies to reduce the data explosion and sparsity problems, including compression and indexes techniques, partitioning and preliminary aggregations. We also examine the relation between compound growth factor and input data density in several industrial cubes.

The paper is organized as following. Section 2 and 3 outlines multidimensional view of data and different OLAP architectures. Section 4 describes sparsity and data explosion phenomenon. Section 5 presents common methods that OLAP sever applies to reduce the data explosion and sparsity problems. Finally we make same conclusions.

2 MULTIDIMENSIONAL VIEW OF DATA IN OLAP SYSTEMS

During the 1990s, a new type of data model, the multidimensional data model, has emerged that has taken over from the relational model when the objective is to analyze data, rather than to perform on-line transactions. Multidimensional data models are designed expressly with the purpose of supporting data analysis. A number of such models have been proposed by researchers from academia and industry. In academia, formal mathematical models have been proposed, while the industrial proposals (Vassiliadis & Sellis 1999, Abelló & Samos & Saltor 2000) have typically been more or less implicitly specified by the concrete software tools that implement them (Pedersen & Jensen 2001).

In a multidimensional data models, there is a set of numeric measures (facts) that are the objects of analysis. Each of the numeric measures depends on a set of dimensions, which provide the context for the measure. For example, the dimensions associated with a sale amount can be the store, product, and the date when the sale was made. The dimensions together are assumed to uniquely determine the measure. Thus, the multidimensional data views a measure as a value in the multidimensional space of dimensions. Often, dimensions are hierarchical; time of sale may be organized as a day-month-quarter-year hierarchy, product as a product-category-industry hierarchy (Chaudhuri &Dayal 1997).

Multidimensional models lie at the core of OnLine Analytical Processing (OLAP) systems. Such systems provide fast answers for queries that aggregate large amounts of detail data to find overall

trends, and they present the results in a multidimensional fashion, which renders a multidimensional data organization ideal for OLAP. Sometimes OLAP and data warehousing are used as synonymous terms. Thomsen (1997) considers them complementary in that data warehousing makes raw data available to end users and ensures its accuracy and consistency (Inmon 1992) whereas OLAP focuses on the end user's analytical requirements (Niemi &Hirvonen & Järvelin 2003).

Actually a data warehouse is a large repository of data integrated from several sources in an enterprise for the specific purpose of data analysis and decision support. Decision support usually requires consolidating data from many heterogeneous sources: these might include external sources in addition to several operational databases. The different sources might contain data of varying quality, or use inconsistent representations, codes and formats, which have to be reconciled. In data warehouses historical and summarized data is more important than detailed. Many organizations want to implement an integrated enterprise warehouse that collects information about all subjects spanning the whole organization (Chaudhuri et al. 1997).

3 OLAP STORAGE OPTIONS

There are two types of OLAP engines: Relational OLAP (ROLAP) and Multidimensional OLAP (MOLAP). In ROLAP, data is itself stored in a relational database whereas with MOLAP, a large multidimensional array is built with the data (Kaser & Lemire 2003). One difficulty with MOLAP is that the array is often sparse. These typically include provisions for handling sparse arrays, and they apply advanced indexing and hashing to locate the data when performing queries. ROLAP systems employing also specialized index structures, such as bit-mapped indices, to achieve good query performance (Pedersen et al. 2001). Generally, MOLAP systems provide faster query response times, while ROLAP systems scale better in the number of facts, are more flexible with respect to cube redefinitions, and provide better support for frequent updates. The virtues of the two approached are combined in the Hybrid OLAP (HOLAP) approach, which stores higher-level summary data using MOLAP technology, while using ROLAP systems to store the detail data (Pedersen et al. 2001).

4 SPARSITY AND DATA EXPLOSION

In OLAP cube, cross product of dimensional members forms the intersections for measure data. But in reality most of the intersections will not have data. This leads to sparsity. Input data or base data (i.e. before calculated hierarchies or levels) in OLAP applications is typically sparse (not densely populated) (Potgieter 2003).

Let’s imagine that we are working for an insurance company with about 100 000 clients. For the needs of marketing analyses we model a cube with 4 dimensions: time, services, offices and sales channels and one measure: number of clients. We have 100 services, 87 offices and 6 channels on the base level. In the upper hierarchical levels we have 12 service categories, 17 regions and 3 channel categories. Our input space consists of 100*87*6 = 52 200 points for a specific moment of time. It is possible some customers to use many services (offered in different offices), but most of the customers use no more than 2 services. So we have maximum 200 000 instances distributed over 52 200 points. We add a new dimension that is important for our analysis – client state with 6 members (new, renewed, lost etc.). Our input space grows to 313 200 points. But the number of clients is still the same and every client has exactly one state, so our 200 000 client were broken to small pieces. We add a new dimension – client gender, and another one – client education, and others – marital status, client occupation etc. From the business point of view these new dimensions are natural characteristics that are subject of interest during the analyses process. But they have significant impact on the size of the multidimensional cube and the sparsity of the data (especially of the input data.

A simple measure for sparsity is the ratio of empty cells compared to all cells (Niemi & Nummenmaa & Thanisch 2000). If we have 60 stores, 500 products, 70 000 customers and 12 months in a year, our

cube has a potential 60×500×70000×12 = 25 200 000 000 cells, but we might only have 360 000 000 non-empty cells in our measure (40 000 customers shopping 12 months a year, buying average on 25 products at 30 stores) making our cube 1.42% dense.

To provide a practical baseline expectation for sparsity in Potgieter (2003) examines data sparsity in a variety of models with a sample of seven companies. The research shows that:

- data density in all cases is significantly less than 1 percent – i.e., extremely sparse;
- as the number of dimensions increases, so does the sparsity of the data (models reviewed have between 5 and 16 dimensions);
- extreme sparsity exists in all industry-specific models (all models have density of less than 1 billionth of a percent).

Truly, our researches over five companies with different branch of business confirm these observations.

Any multi-dimensional model needs to provide space for every possible combination of data points. Since in sparse models most data points are zeros, the main issue is how to store all values other than zero values. For example, if the data density of a model is 1% and there is no sparsity handling, the resulting model will be 100 times larger than a model that has perfect sparsity handling. Sparsity handling is the efficient storage of very sparse data (Potgieter 2003).

Data explosion is the phenomenon that occurs in multidimensional models where the derived or calculated values significantly exceed the base values. There are three main factors that contribute to data explosion (Potgieter 2003):

- sparsely populated base data increases the likelihood of data explosion;
- many dimensions in a model increase the sparsity and therefore data explosion;
- a high number of calculated levels in each dimension increase the likelihood of data explosion;

The multidimensional cross relationships which exist in all OLAP applications and the fact that the input data is usually very sparse are the main reason most of the olap applications to suffer from data explosion consequences. The thinly distributed input data values may each have literally hundreds of computed dependent cells, because they will feature in hierarchies in each of the dimensions. This means that the 'computed space' is much denser than the input data. The result is that pre-computed results based on sparse multidimensional data are far more voluminous than might be expected; this is independent of the storage technology used (Pendse 2005).

In (Pendse 2005) the compound growth factor (CGF) was introduced as a measure for data explosion. According (Pendse 2005) if the database grew by a ratio of 5.83 with two dimensions, the growth factor is 2.4 (the square root of 5.83) per dimension. So we can adopt that

$$\mathrm{CGF} = \sqrt[n]{\frac{Ag + Bs}{Bs}}$$

where n is a number of dimensions, Bs is a number of non empty base cell, Ag is a number of non-empty consolidated cells.

With large dimensions, there will usually be more levels of consolidation so the CGF is also likely to be higher. It also reduces if dimensions have few derived members or if data is clustered rather than being randomly distributed (because fewer consolidated members will become populated with very sparse data) .

5 COMMON METHODS TO REDUCE THE DATA EXPLOSION AND SPARSITY PROBLEMS

Because the extremely sparse cubes are frequent phenomenon, OLAP engines offer different methods of increasing the performance and reducing the size of the cubes. Naturally the storage technologies (rolap or rolap) determine the appropriate techniques and the particular database management system determines their availability and variety, but nevertheless many of them are similar.

5.1 Online or preliminary calculation

OLAP data explosion is the result of multidimensional preaggregation. Data that has not been preaggregated is not available for reporting and analysis purposes unless calculated at run time. Although nowadays the disk space is cheap the full preaggregation materializing all combinations of aggregates is still infeasible - it takes too much storage and initial computation time (Pendse 2005, Pedersen et al. 2001, Pederseny & Christian & Curtis & Dyreson 1999).

Most Molap server tried to determines which aggregations provide the greatest performance improvements or offer advisors to produces recommendations. The advisors analyze the OLAP metadata model and/or data snapshots and tried to determine the optimum set of aggregations from which all other aggregations can be derived. Molap servers also enable the database administrator to make a trade-off between system speed and the disk space required to manage aggregations.

For example the Usage-Based Optimization Wizard in Microsoft OLAP Services enables the database administrator to instruct OLAP Services to create a new set of aggregations for all queries that take longer than n seconds to answer (Vitt 2007).

Oracle Olap Option supports two aggregation methods: cost-based and level-based. With cost-based aggregations the administrator can specify the percentage of data to be precomputed. Level-based aggregations enable the administrator to specify which level should be stored (those levels that are most commonly viewed by the end users).

In Rolap environment the problem with proper selection of preliminary aggregation is well known as view selection problem (Talebi & Chirkova & Fathi 2009). Many approaches to the problem have been explored in academia researches, including greedy algorithms, randomized search, genetic algorithms and others (Morfonios & Konakas & Ioannidis & Kotsis 2007). The editors of RDBMS offer advisors for recommendations of summary tables or materialized views creation.

However, the Molap servers offer better summary management facilities. The Relational OLAP approach starts off with the premise that OLAP queries can generate the multidimensional projections on the fly without having to store and maintain them in foreign storage. But in large data warehouses, indexing alone is often not sufficient to achieve high performance for some queries. In this case, the ROLAP approach relies on selecting and materializing in summary tables the "right" subsets of aggregate views along with their secondary indexing that improves overall aggregate query processing (Baralis & Paraboschi & Teniente 1997, Gupta & Harinarayan & Rajaraman & Ullman 1997, Gupta 1997). Like the MOLAP case, controlled redundancy is introduced to improve performance. But straight forward relational storage implementation of materialized ROLAP views and B-tree indexing on them is wasteful on storage and inadequate on query performance and incremental update speed (Kotidis & Roussopoulos 1998).

RDBMS editors also consider these facts. They tried to support and develop hybrid online analytical processing. For example in the last 11g version Oracle OLAP options has new features designed to make easy to use the cube as a summary management solution for applications that query relational tables (summary relational tables are transparently substituted with olap cubes) (Oracle Documentation 2010).

5.2 OLAP Indexing

Indexes are used to speed up queries in combination with data aggregates. Since both consume the same resource - space, their selection should be done together for the most efficient use of space and query speed optimization.

Database indexes provided today by most relational database systems use B+-tree (value-list Indexes) indexes to retrieve rows of a table with specified values involving one or more columns. The leaf level of the B-tree index consists of a sequence of entries for index key values. Each key value reflects the value of the indexed column or columns in one or more rows in the table, and each key value entry references the set of rows with that value (O'Neil & Quass 1997).

Another common used index type in RDBMS is a bitmap index. Bitmap indexes are known as the most effective indexing methods for range queries on append-only data, and many different bitmap indexes have been proposed in the research literature (Wu at al. 2010, Bookstein & Klein 1990, Hu & Sundara & Chorma & Srinivasan, 2005, Lin & Li & Tsang 1999, O'neil & O'neil & Wu 2007). An important consideration for database query performance is the fact that Boolean operations, such as AND, OR, and NOT are extremely fast for Bitmaps. To optimize Bitmap index access, Bitmaps can be broken into Fragments of equal sizes to fit on single fixed-size disk pages. Corresponding to these Fragments, the rows of a table are partitioned into Segments, with an equal number of row slots for each (O'Neil at al. 1997).

However, most commercial implementations are relatively simple, primarily the basic bitmap index (O'Neil 1987) and the bit-sliced index (O'Neil at al. 1997). There are a significant number of promising techniques proposed in the research literature, but they have not gained wide acceptance. Wu at al. 2010 believe that a poor understanding of their performance characteristics is the most important reason for this lack of acceptance. This motivated them to conduct a thorough analysis of bitmap indexes performance characteristics.

In Molap environments b-tree and hash index structure over cube dimensions are used to index into the blocks of data stored as multi-dimensional arrays. Some molap servers use composites as a compact format for storing sparse multidimensional data. The composite holds only the combinations of the sparse dimensions that actually contain data. For an extremely sparse data a compressed composites are provided.

5.3 OLAP Compression algorithms

Compression algorithms are used to compress the OLAP data, so that the data-size can be manageable. Given the issue of data-explosion and the typical large size data needs of OLAP, this is a big benefit.

There are many compression algorithms available and almost all OLAP servers do use the compression algorithms.

For example Essbase allows you to choose whether data blocks that are stored on disk are compressed, as well as which compression scheme to use - bitmap compression, Run-length encoding, zlib compression or Index Value Pair compression. When data compression is enabled, Essbase compresses data blocks when it writes them out to disk (Essbase Documentation 2010).

Oracle Database 10g introduced patented compressed cube technology that optimized the aggregation, storage and query of the sparse data sets. The compressed cube technology takes advantage the sparsity patterns in hierarchical data sets by mapping more than one cell within a node of a hierarchy to a single stored value rather than replicating that same value up the hierarchy. The result is faster aggregation, more efficient storage on disk and faster queries (Oracle Corporation 2008).

5.4 OLAP Partitioning

Whenever a database handles volumes of data, the concept of data partitioning across multiple data storage becomes important. In a data warehouse environment, for example, partitioning allows huge tables to be broken up into series of smaller tables with faster access, but the SQL application can query the series of smaller tables as if it were one big table. With partitioning, a data warehouse can expand to many hundreds of terabytes, while ensuring that results are returned in a reasonable amount of time. A partitioned architecture also enables you to drop a partition, which makes less than a second. If you do not partition, deleting old data could take a long time (Schrader & Vlamis &Nader & Collins & Claterbo & Campbell & Conrad 2010).

OLAP cubes can also benefits from partitioning data. By partitioning data, the cube can be broken into more manageable chunks, but the complexity of this strategy is hidden from the application and end user.

Oracle OLAP and Essbase both offer partitioning strategies. Oracle OLAP allows the cube designers to define a dimension as a partitioning dimension. Generally, the designers identify a level (such a year) of the partitioning dimension, although more complex designs are possible. Cubes are physically separated into individual partitions by year, but the partitioning scheme is transparent to applications that query or write to a cube. At query time, partition pruning occurs – meaning if the cube is partitioned by year and the user query ask for data for a single year, only one partition is accessed for the data. Partitions can be added and removed easily.

In Essbase, the partition is a region of a cube that is shared with another cube. Partitions come in there types: Replicated partitions allow you to store data in different cubes and copy shared data form one cube to another; Transparent partitions enable users to navigate seamless form locally stored data to remotely stored data; Linked partitions provide a means of linking a cell in a cube to a different cube with potentially different dimensionality (Schrader et al.2010).

6 CONCLUSION AND FURTHER WORK

In this paper we discuss a data explosion and sparsity phenomenon in online analytical systems which are closely related with the performance issues in olap applications. Our investigations verified the observations that Johann Potgieter has made in (Potgieter 2003) - extreme sparsity (less than 1 percent) are very common; they exists in all industry-specific models that we have met. We also take a view of common explosion handling methods as indexes, partitioning and compression and preliminary aggregations. Proper aggregations are very important in achieving OLAP browsing speed. But this is a non-trivial (NP-complete) optimization problem with many influencing factors: space use, update speed, response time demands, actual queries, prioritization of queries and others. Almost all DBMSes (ROLAP+MOLAP) now use similar, but more advanced, techniques for determining best aggregates to materialize and optimize the query performance (Pedersen 2006).

In spite of all according to independent olap survey (Pendse 2007) more and more people are reporting query performance as a serious problem. In fact, it is the only problem that has gotten worse every year. This is a curious result because that hardware gets ever faster and cheaper, and most software vendors boast that each new release has been tuned to perform better than before. The most obvious theory is that rocketing data volumes are the cause. If data volumes are increasing even faster than hardware performance improves, it might explain the rising complaints about performance. But looking at the actual median query times and comparing them with the median input data volumes over the last five years, the olap survey gets some more surprises - there is no evidence of exploding data volumes. The ratio of query time in seconds to input data volumes in gigabytes has stayed constant at 1.7 sec/GB for the years between 2004 and 2007, and it is higher than the 1.5 sec/GB in

2003. Nigel Pendse supposes that the undoubted improvements in hardware performance have been absorbed by less efficient software. For example, the move to thin-client and multilayered architectures may have many advantages, but better performance is not necessarily one of them. Good old-fashioned client/server architectures, if implemented well (as they are in the best OLAP tools), are very hard to outperform. But we prone to think that in the last years the end user of OLAP applications are more experienced so their requirements and expectations are grow up. Also the more powerful software platforms and physical resourses tempt the olap applications designers and developer to implement more general-purpose models and to satisfy more business analyst requests.

On the base observed statistical data the author of olap survey put the following question: “Over the last few years, we've all become used to free, near-instant searches of unimaginably large indexes of tens of billions of Web pages, images, videos, news channels, Usenet postings and books; if a PC bought today is around ten times as fast as a more expensive model from as little as five years ago, why haven't our BI applications speeded up by a similar factor?”

This question stays open for answering. But here we can find the answers why the performance of olap systems still excited the academia researchers and worried the applications end users.

Acknowledgements: This paper is supported by the Sofia University SRF under Contract 163/2010.

References

Plantevit, M., Laurent, A., Laurent, D., Teisseire, M. and Choong , Y (2010). Mining Multidimensional and Multilevel Sequential Pattern, ACM Transactions on Knowledge Discovery from Data, vol.4 , Issue 1, No. 4.

Wu, K., Shoshani, A. and Stockinger, K. (2010). Analyses of Multi-Level and Multi-Component Compressed Bitmap Indexes, ACM Transactions on Database Systems, Volume 35, Issue 1, Article No.2.

Wu, S., Jiang, Sh., Ooi, B. and Tan K. (2009). Distributed online aggregations, Proceedings of the VLDB, vol. 2, Issue 1.

Bellatreche, L. and Woameno, Y. (2009). Dimension Table driven Approach to Referential Partition Relational Data Warehouses, DOLAP '09: Proceeding of the ACM twelfth international workshop on Data warehousing and OLAP.

Pendse, N. (2005). Database explosion, Business Application Research Center.

Pendse, N. (2008). What is OLAP?, Business Application Research Center.

Vassiliadis, P. and Sellis, T. (1999). A Survey of Logical Models for OLAP Databases, ACM SIGMOD Record, vol.28, no. 4, , pp. 64-69.

Abelló, A., Samos, J., and Saltor, F. (2000). A Data Warehouse Multidimensional Data Models Classification, Technical Report.

Pedersen, T. and Jensen, Ch. (2001). Multidimensional Database Technology, IEEE Computer Society Press, vol. 34, Issue 12, pp.40 - 46, ISSN:0018-9162.

Chaudhuri S.and Dayal., U. (1997). An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26:65-74.

Thomsen, E. (1997). OLAP Solutions Building Multidimensional Information Systems. John Wiley and Sons, USA.

Inmon, W. H. (1992). Building the Data Warehouse. QED Technical Publishing Group, Wellesley, Massachusetts.

Niemi, T., Hirvonen, L. and Järvelin, K. (2003). Multidimensional data model and query language for informetrics, Journal of the American Society for Information Science and Technology , Volume 54 Issue 10.

Kaser, O. and Lemire, D. (2003). Attribute Value Reordering for Efficient Hybrid OLAP, Proceedings of the 6th ACM international workshop on Data warehousing and OLAP,pp.1-8,ISBN:1-58113-727-3.

Potgieter, J. (2003). OLAP Data Scalability, DM Review Magazine, October 2003. Available at: http://www.dmreview.com/dmdirect/20031031/7636-1.html.
Niemi, T., Nummenmaa J. and Thanisch P. (2000). Functional Dependencies in Controlling Sparsity of OLAP Cubes, Springer Berlin / Heidelberg, Volume 1874/2000, pp.199-209
Pederseny, T., Christian, S. Curtis, J. and Dyreson, E. (1999). Extending Practical Pre-Aggregation in On-Line Analytical Processing, VLDB'99, pp.663-674.
Talebi, Z., Chirkova, R. and Fathi, Y. (2009). Exact and inexact methods for solving the problem of view selection for aggregate queries, International Journal of Business Intelligence and Data Mining, Volume 4, Issue 3/4, pp.391-415.
Morfonios, K., Konakas, S., Ioannidis, Y. and Kotsis, N. (2007).ROLAP Implementations of the Data Cube, Computing Surveys (CSUR) , Volume 39 Issue 4.
Baralis, E., Parabosch,i S. and Teniente, E. (1997). Materialized View Selection in a Multidimensional Database, In Proc. of the 23th International Conference on VLDB, pp. 156-165, Greece.
Gupta, H., Harinarayan, V., Rajaraman, A. and Ullman, J. (1997). Index Selection for OLAP, In Proc. of the Intl. Cunf on Data Engineering, pp. 208-219, UK.
Gupta, H. (1997). Selections of Views to Materialize in a Data Warehouse, In Proceedings of ICDT, pp.98-112, Delphi.
Kotidis, Y. and Roussopoulos, N. (1998). An Alternative Storage Organization for ROLAP Aggregate Views Based on Cubetree, SIGMOD '98, Volume 27, Issue 2.
Essbase Documentation (2010). Database Administrator's Guide, Available at: http://download.oracle.com/docs/cd/E12825_01/nav/portal_3.htm
Oracle Documentation (2010). Available at: http://www.oracle.com/pls/db112/homepage
[Oracle Corporation (2008). New Features and Performance Advances in the OLAP Option to the Oracle Database 10g , Technical whitepaper, Available at: http://www.oracle.com/technology/products/bi/olap/index.html
O'Neil, P. and Quass, D. (1997). Improved Query Performance with Variant Indexes, SIGMOD 1997, pp. 38-49, USA.
O'Neil, P. (1987). Model 204 architecture and performance, In Proceedings of the 2nd International Workshop in High Performance Transaction Systems, Lecture Notes in Computer Science,vol. 359. Springer, pp.40–59.
Pendse, N. (2007). The Problems with OLAP, Information Management Magazine, March 2007
Schrader, M., Vlamis, D., Nader, M., Collins, D., Claterbo,s Ch., Campbell, M. and Conrad, F. (2010). Oracle Essbase & Oracle OLAP , Publisher The McGraw-Hill Companies, Inc., ISBN 978-0-07-162183-3.
Pedersen, T. (2006). DW Performance Optimization, Aalborg University Lections, Available at: www.cs.aau.dk/~tbp/Teaching/DWML06/6_DWPerformance.pdf
Bookstein, A. and Klein, T. (1990). Using bitmaps for medium sized information retrieval systems, Inform. Process. Manag. 26, pp.525–533.
Hu, Y., Sundara, S., Chorma, T. and Srinivasan, J. (2005). Supporting RFID-based item tracking applications in Oracle DBMS using a bitmap datatyp., In Proceedings of the International Con ference on Very Large DataBases (VLDB), pp. 1140–1151.
Lin, X., Li, Y. and Tsang, P. (1999). Applying on-line bitmap indexing to reduce counting costs in mining association rules., Inform. Sci. 120, 1-4, pp. 197–208.
O'neil, E., O'neil, P. and Wu, K. (2007). Bitmap index design choices and their performance implications. In Proceedings of IDEAS'07. pp. 72–84.
Vitt, E. (2007). Microsoft SQL Server 2005 Analysis Services Performance Guide, White Paper, White Paper.